**Processing Lists: builtins**

- `sum(x)` adds up all the elements in the list `x`  
  - They must all be numbers!
- `min(x)` or `max(x)` find the min/max value in list `x`  
  - They use the same ordering as `sort()`
- `range(a,b,c)` produces `[a,a+c,a+2c,…,a+c*(b-a)/c]`  
  - Starts at `a`, increases by `c` each time, until `b` (or less)
  - The argument `c` is optional; `c = 1` by default
- `list(x)` converts `x` (such as a string) to a list  
  - Example: `list('mimsy')` produces `['m', 'i', 'm', 's', 'y']`

**The Map Function**

- `map(⟨function⟩, ⟨list⟩)`  
  - Function has to have exactly 1 parameter
  - Otherwise, get an error
  - Returns a new list  
  - Does the same thing as `def map(f,x):`  
    - `result = [] # empty list`
    - `for y in x:`
      - `result.append(f(y))`
    - `return result`
  - `map(f, x) [f(x[0]), f(x[1]), …, f(x[n-1])]`  
    - Calls the function `f` once for each item

**Two Dimensional Lists**

- **Table of Data**
  - **Images**
  - Each row, col has a value
  - Each row, col has an RGB value
  - Store them as lists of lists (row-major order)
  - `d = [[5,4,7,3],[4,8,9,7],[5,1,2,3],[4,1,2,9],[6,7,8,0]]`

**Overview of Two-Dimensional Lists**

- **Access value at row 3, col 2:**
  - `d[3][2]`
- **Assign value at row 3, col 2:**
  - `d[3][2] = 8`
- **An odd symmetry**  
  - Number of rows of `d`: `len(d)`  
  - Number of cols in row `r` of `d`: `len(d[r])`

**How Multidimensional Lists are Stored**

- `b = [[9, 6, 4], [5, 7, 7]]`
  - `b` holds name of a one-dimensional list  
    - Has `len(b)` elements
    - Its elements are (the names of) 1D lists
  - `b[i]` holds the name of a one-dimensional list (of ints)  
    - Has `len(b[i])` elements

**Image Data: 2D Lists of Pixels**

- `b[0][0]` is a white pixel
Slices and Multidimensional Lists

- Only “top-level” list is copied.
- Contents of the list are not altered

\[ b = [[9, 6], [4, 5], [7, 7]] \]

```python
x = b[0]
```